Reflections on SDSI

Ronald L. Rivest

Vannevar Bush Professor of EECS
MIT, Cambridge, MA

Lampson Fest

February 13, 2014

Context 1970–1990

- Invention of public-key cryptography (Diffie & Hellman 1976, RSA 1977)

Context 1970–1990

- Invention of public-key cryptography (Diffie & Hellman 1976, RSA 1977)
- Invention of public-key certificate (Kohnfelder, MIT B.S. thesis, 1978). Binds name to public key.

Context 1970–1990

- Invention of public-key cryptography (Diffie & Hellman 1976, RSA 1977)
- Invention of public-key certificate (Kohnfelder, MIT B.S. thesis, 1978). Binds name to public key.
- X.509 hierarchical public-key infrastructure and certificates (1988). Envisions strict hierarchy of certificate authorities.

Context 1970–1990

- Invention of public-key cryptography (Diffie & Hellman 1976, RSA 1977)
- Invention of public-key certificate (Kohnfelder, MIT B.S. thesis, 1978). Binds name to public key.
- X.509 hierarchical public-key infrastructure and certificates (1988). Envisions strict hierarchy of certificate authorities.
- Invention of World Wide Web (TBL, first browser 1990) – causing explosive growth of digital communications and e-commerce.

Inquiring minds wanted to know:

- How can public-key technology best be used to secure the Internet?

Inquiring minds wanted to know:

- How can public-key technology best be used to secure the Internet?
- Can’t we invent something simpler and better than X.509 and ASN.1 ?

Inquiring minds wanted to know:

- How can public-key technology best be used to secure the Internet?
- Can’t we invent something simpler and better than X.509 and ASN.1 ?
- What do we really need?

Inquiring minds wanted to know:

- How can public-key technology best be used to secure the Internet?
- Can’t we invent something simpler and better than X.509 and ASN.1 ?
- What do we really need?
- What’s in a name?

Inquiring minds wanted to know:

- How can public-key technology best be used to secure the Internet?
- Can’t we invent something simpler and better than X.509 and ASN.1 ?
- What do we really need?
- What’s in a name?
- Do we really need CRL’s?

Inquiring minds wanted to know:

- How can public-key technology best be used to secure the Internet?
- Can't we invent something simpler and better than X.509 and ASN.1 ?
- What do we really need?
- What's in a name?
- Do we really need CRL's?
- ...much discussion and unhappiness with existing framework and tools...

SPKI begins

- Feb ’96: Perry Metzger begins *SPKI (Simple Public Key Infrastructure)* mailing list.

SPKI begins

- Feb ’96: Perry Metzger begins *SPKI (Simple Public Key Infrastructure)* mailing list.
- Carl Ellison gives many “use cases” not yet well handled, such as granting of permissions.

SPKI begins

- Feb ’96: Perry Metzger begins *SPKI (Simple Public Key Infrastructure)* mailing list.
- Carl Ellison gives many “use cases” not yet well handled, such as granting of permissions.
- Inspired by earlier work by Lampson, Ellison also argues for elimination of names in favor of using public-keys as the *only* handles (identifiers) for principals.

Lampson/Rivest start thinking about names

- March 1996: Lampson and Rivest begin discussions with Ellison, other SPKI folks, and with each other, on these issues, especially names.

Lampson/Rivest start thinking about names

- March 1996: Lampson and Rivest begin discussions with Ellison, other SPKI folks, and with each other, on these issues, especially names.
- Lampson emails (1 mar 96):
  “So my belief is that anything people have to look at should be stated in terms of meaningful names, not keys. The keys should be kept internal to the system. Of course you can say that you’ll have extra certificates linking names to keys, but the names will still be the "real" thing. It’s true that the system takes action based on messages being signed by keys, but the configuration, which is the important thing, is established in terms of names, since that’s the only way people can describe it. So it must be that the names are the real thing and the keys just an internal mechanism. ”

Names

Names want simultaneously to be:

- Short, memorable.

Names

Names want simultaneously to be:

- Short, memorable.
- Meaningful and easy to use.

Names

Names want simultaneously to be:

- Short, memorable.
- Meaningful and easy to use.
- Local (non-hierarchical; bottom-up).

Names

Names want simultaneously to be:

- Short, memorable.
- Meaningful and easy to use.
- Local (non-hierarchical; bottom-up).
- Globally unique.

Names

Names want simultaneously to be:

- Short, memorable.
- Meaningful and easy to use.
- Local (non-hierarchical; bottom-up).
- Globally unique.

These are not compatible!

It gets worse!

- If PKI and certificates are mostly about bindings of names to public keys: how do you know *who is authorized* to assert such a binding for a given name?

It gets worse!

- If PKI and certificates are mostly about bindings of names to public keys: how do you know *who is authorized* to assert such a binding for a given name?
- Especially if names are non-hierarchical?

It gets worse!

- If PKI and certificates are mostly about bindings of names to public keys: how do you know *who is authorized* to assert such a binding for a given name?
- Especially if names are non-hierarchical?
- Who is relevant “CA” for a name?

SDSI

Lampson and Rivest publish draft SDSI (Simple Distributed Security Infrastructure) in June 1996:

- Innovation: *Associate a name space with each public key*.

SDSI

Lampson and Rivest publish draft SDSI (Simple Distributed Security Infrastructure) in June 1996:

- Innovation: *Associate a name space with each public key*.
- In effect, each name now has the form of a dotted pair consisting of a public key and an identifier.

PK.identifier

SDSI

Lampson and Rivest publish draft SDSI (Simple Distributed Security Infrastructure) in June 1996:

- Innovation: *Associate a name space with each public key*.
- In effect, each name now has the form of a dotted pair consisting of a public key and an identifier.

$$PK.identifier$$

- PK is the only PK authorized to sign bindings for *PK.identifier*. Certificate thus has form:

$$\text{PK.identifier} \Longrightarrow \text{PK' (signed by PK)}$$

SDSI Advantages

- Names (identifiers) can be local and meaningful to issuer.

SDSI Advantages

- Names (identifiers) can be local and meaningful to issuer.
- Name conflicts avoided; global uniqueness OK.

SDSI Advantages

- Names (identifiers) can be local and meaningful to issuer.
- Name conflicts avoided; global uniqueness OK.
- Names can naturally refer to *groups*.

SDSI Advantages

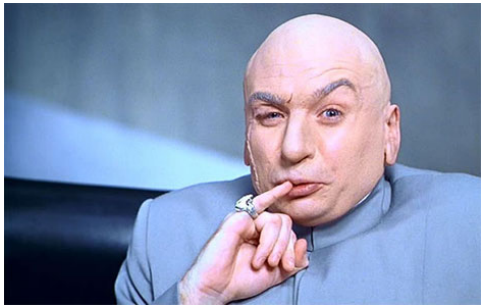
- Names (identifiers) can be local and meaningful to issuer.
- Name conflicts avoided; global uniqueness OK.
- Names can naturally refer to *groups*.
- *Extended names* have a nice algebra:

  $$PK.Microsoft.Research.ButlerLampson$$

  chains four name spaces together to give nice indirect handle for Butler, even if I only know public key of Microsoft; Certificate can bind to extended name:

  $$PK.butler \Longrightarrow PK.Microsoft.Research.ButlerLampson$$

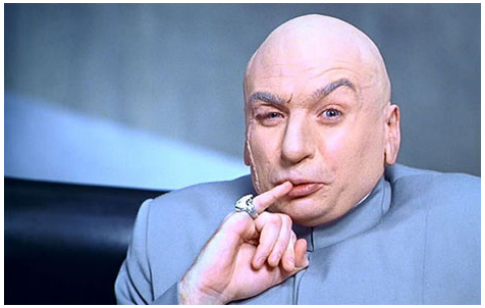
What happened to World Domination?



Why didn't SDSI take over?

- SDSI is great for writing ACL's—oriented more towards access-control than for authentication.

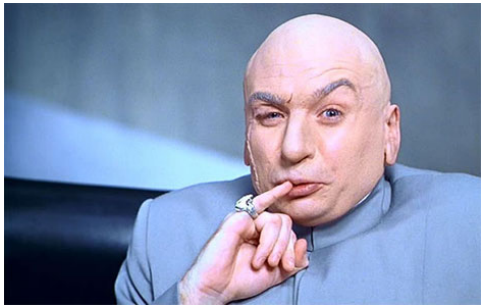
What happened to World Domination?



Why didn't SDSI take over?

- SDSI is great for writing ACL's—oriented more towards access-control than for authentication.
- Elegant naming algebra still leaves an interesting (but solvable) search problem for finding certificate chains. This starts with (requesting) key, and finds explanation why it is implied by ACL.

What happened to World Domination?



Why didn't SDSI take over?

- SDSI is great for writing ACL's—oriented more towards access-control than for authentication.
- Elegant naming algebra still leaves an interesting (but solvable) search problem for finding certificate chains. This starts with (requesting) key, and finds explanation why it is implied by ACL.
- In practice, search problem is often vaguer: given attributes of another principal, find their public key.

Thanks and Happy Birthday, Butler!